12th IMEKO TC1 & TC7 Joint Symposium on
Man Science & Measurement
September, 3 – 5, 2008, Annecy, France

CONSTRAINTS DRIVEN SUBSCRIPTION USING CONFIDENCE INTERVAL MODEL

Jean-Luc Sarrade[1], *Stéphane Perrin*[2]

[1] Soft Formation Corp., Alex, France, jean-luc.sarrade@univ-savoie.fr
[2] LISTIC, Univsersité de Savoie, Annecy, France, stephane.perrin@univ-savoie.fr

Abstract: In the context of intelligent instrument network, i.e. intelligent sensors and actuators, the user or the designer has to manipulate variable network functions. Service model allows the designer to formalize and manipulate information access. Service model also permits to, check, control and react from instrument data and this approach is web-enable. Producer-Consumer pattern facilitates design and intelligent instrument networking. The connection between producer and consumer can be checked (i.e. type of data) and filtered (condition of sending). We propose to extend the producer-consumer approach using service model for variable subscription service where conditions of subscription constraint could be expressed. We consider interval confidence model of data and several consumers to produce the event or data to be subscribed. A prototype is presented to illustrate and validate the approach on the OSGi platform.

Keywords: subscription, constraint, interval confidence

1. INTRODUCTION

Intelligent instruments, i.e. intelligent sensors and actuators, are now commonly used in industry and home automation [1][2]. Recent studies [3][4][5] discuss their design and model. Intelligent instruments have to exchange measurement information. To support them, variable network is generally used when using fieldbus.

In a context of intelligent instrument network, when using fieldbus network, variable network model is native inside the protocol. So associated functionalities from variable network are usually offered by the network protocol, like Echelon network or CAN network for basics uses. According to the filedbus protocol, the initiative access can come from the producer (push) or the consumer (pull). Generally, these functionalities for variable network management are limited to access mode, i.e write or read, and the fieldbus protocol deals with variable updating process.

The generalization of Internet supports low cost interconnection solutions/gateway for various network system - wireless or not: Zig-Bee, Bluetooth, RFID, Echelon, CAN, X10, etc… - needs the same kind of functionalities.

In a context of no-filedbus networks, like Internet - TCP-IP protocol, the variable access functionality can be obtain by service modeling. The use of service model facilitates intelligent instrument networking and allows producer/consumer pattern [6]. Recent studies propose component model based on producer/consumer pattern [7]. In addition, service model approach is web-enable.

Due to its process capability, intelligent instrument is able to provide various functions: corrected value, signal processing, etc… Network variable access can be considered as service in a non-fieldbus network. In addition, service model can be support dynamic functionalities: service detection, plug-and-play capability, etc…

We suppose to use interval confidence model to represent information in order to improve knowledge of information and so, to improve decision process.

Information is supposed provided from sensors or fusion process. The entity providing the results of fusion process from one or several producer can be considered as a producer. Coupled with filtering functionality, this producer is able to provide subscription services.

This paper presents a solution to give the possibility to the user (or a consumer) to request a subscription service dynamically. User (or a consumer) has to be able to generate a new subscription service by giving the description of sources (consumers) and the aggregation/fusion process.

Next section presents the architecture of a solution for subscription process in a consumer/producer context.

2. SUBSCRIPTION FACILITIES

2.1. Introduction and context

The context of our studies is based on consumer producer model. This approach allows the user or designer of application to focalized on information to be exchange, the communication protocol is dealt with the consumer/producer model implementation.

Our proposition consists to propose an architecture model where creating dynamically an aggregation/fusion process from one or several sensors is possible. In addition, a consumer has the possibility to make subscription process using filter. The filter contains the condition of sending the information to the consumer.

2.2. Producer-consumer

A connector bind the producer P to the consumer C. Connector supports the communication capabilities and could be able to verify the compatibility of exchanged data type. Fig 1 illustrates consumer/producer representation.

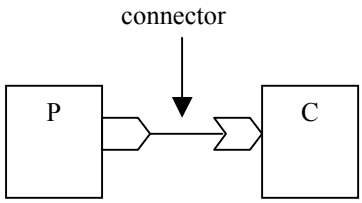

Fig. 1. Producer-Consumer symbol

The connection type could be declined into different kind : event (i.e. alarm) or data (i.e. measure).

The connector or the producer can contain the condition of sending information from the producer. It is usually possible to fix the condition of updating information for the consumer side. This kind of functionality can be expressed when binding (for example: Echelon network binding process or filter using OSGi platform [6]).

Designer of application is able to choice exchange mode access: push, pull and expressed conditions to allow the updating process to take place: threshold, hysteresis, time frequency, etc. By this way, it is possible to implement subscription functionality. But usually, the constraints of subscription either are authorized by the system (for basic subscription constraints) or have to be implemented using specific code to be added into the producer. See Fig 2.

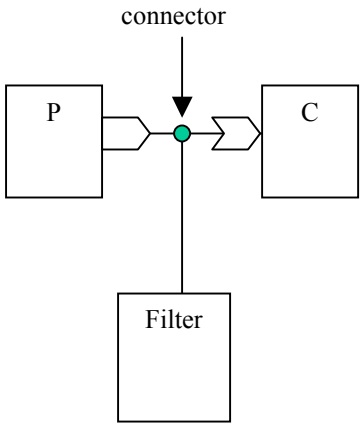

Fig. 2. Filter positioning

2.3. Fusion process

Otherwise, when system is running, a consumer, called C_d, could need new information. This can be computed from known information from several consumers. In this scenario, consumer Cd can express fusion process. Then several scenarios are possible:

- Fusion process is a shared resource (i.e. service) and be evocable from the consumer. In this case, consumer C_d has to be into relation with the provider of fusion process (i.e a service) and be able to transmit parameters. For example, in the case of a weighted arithmetic mean process as fusion process, C_d has to be connected to the fusion process service and transmit parameters (ie identifier of each providers) and weight for each value.
- Fusion process is not available in the system as shared resource. Then, consumer C_d could nevertheless obtain the result of fusion process if it is possible to provide an external fusion code or if a parser exists to provide desired fusion process (i.e external class in an OSGi platform) from a description given by the consumer (XML file for example).

The first scenario is more flexible and OSGi platform allows dynamical update/upload of bundles. Then new fusion process can be (re)loaded into the application during the platform running. This approach could be used to assume the second scenario.

2.4. Information type

The information entity exchanged throw the data flow is defined as below:

$$IE=<ic, id, unit, properties, \text{date}> \quad (1)$$

where:

id is the identifier of this information entity,

ic = { V_{min}, V_{max} } is the confidence interval of the measured or computed values,

unit is the SI unit in which the values V_{min} and V_{ma} are expressed,

properties is a list of properties that contains various semantic information like the nature of data or the location. This list of properties is used to facilitate binding process when *id* of data is not used.

```
properties={
        (name1, value),
        (name2, value),                    (2)
        …
}
```

2.5. Subscription capability

Subscription functionalities providing by systems are usually basic and fixed by the system. They are usually not include interval confidence representation.

In the next section, we propose an architecture which includes confidence interval on filter constraints. We show that the user can create dynamically a new subscription service instance and the associated non-basic filter.

In addition, several sources could be necessary to the fusion process. This can be appeared when system is running. Then the proposed architecture presented in the next section propose to take into consideration the possibility to a consumer to subscribe to an information formed from desired fusion process.

3. FUSION/SUBSCRIPTION ARCHITECTURE

In this section we describe the architecture to allows the user - or a consumer - to dynamically request information from fusion process and expressed the condition of updating (i.e. filter of sending).

3.1. Subscription request

The user or a consumer has to create a subscription entity. Elements to create the required subscription have to be provided into the system. This information contains:

- list of producer,
- identifiers of aggregation/fusion process,
- identifiers and parameters of filter.

This information could be placed into an xml file; fig 3 illustrates consumer's request of subscription.

Note that list of producers could be replaced by yellow page service for example

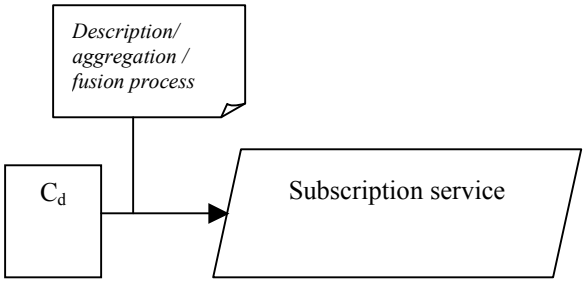

Fig. 3. consumer's request of subscription

3.2. Subscription service steps.

Consumer/user launch the request and then subscription service entity receives the request of user/consumer and the xml description. Here steps for subscription process

- a new (virtual) provider *Sub* is generated.
- Each provider is connected to the Sub entity (virtual sensor), according to the specified constraints (xml description).
- Connection between Sub entity and fusion/aggregation process is done.
- Finally a the virtual provider Sub is connected to the consumer C_d.

The Fig 4 shows a snapshot of configuration after these steps.

3.3. Constraints

The subscription constraints are provided from consumer's request. Some of them could be specified by the system for the most current of them (i.e. threshold, time frequency, etc.) and other could be dynamically provided by the consumer C_d to the provider *Sub*. In the same way, the consumer C_d can also specify the fusion process and some of its internal parameters.

The combining of constraints is also possible.

Example of constraint combining (using interval confidence)

$$(V_{min1}<0.5 \text{ and } V_{max1} > 0.7) \text{ and } ((V_{min2}>3.1 \text{ and } V_{max2} >3.2) \quad (3)$$

3.4. Fusion/aggregation process

Consumer C_d requests to the system the desired fusion process. It also provides parameters. For example, formula (5) defines, the results desired information I_f: computed using weighted average A_w method:

$$I_f = A_w (I_1,w_1, I_2, w_2,..., I_n , w_n) \quad (4)$$

Aw identifier will be used to identify/find the corresponding service of weighted average method in the system. In [8], author explains how propagate interval confidence; results can be used to compute the weight average.

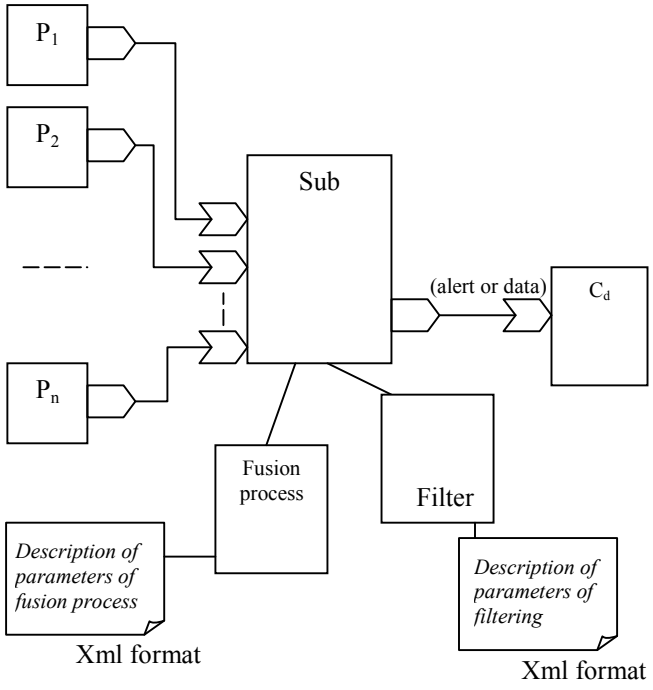

Fig. 4. snapshot of configuration after subscription process

3.5. Fusion/aggregation information and subscription

Consumer's request could contain both fusion method identifier and (filter) subscription constraints.

$$I_f = A_w (I_1,w_1, I_2, w_2,..., I_n , w_n) \quad (5)$$

$$I_{fmoy}(1-5\%) < I_f < I_{fmoy}(1+5\%) \quad (6)$$

In (5), fusion process is defined and in (6) the condition of sending value is defined. I_f is send to the consumer when (6) is true.

4. PROTOTYPE

An implementation of the proposed architecture is developed on Open Services Gateway Initiative (OSGi) platform [6]. The OSGi Alliance creates open specifications for the network delivery of managed services to devices in

the home, car, mobile and other environments. The OSGi Service Platform specifications provide all the necessary mechanisms required to be able to manage these services over networked devices. Any device which has the OSGi Service Platform installed on it, can manage the life-cycle of the components in the device along with installing, updating or removing services without disrupting the operation of the system. At the core of the OSGi Service Platform Specification is the OSGi Framework. It provides a general-purpose, secure, and managed Java framework that supports the deployment of bundles.

The OSGi Framework provides the infrastructure to dynamically install, uninstall, update, start and stop bundles. The bundles use the OSGi Framework to register and deregister their services, and also to look up and track other bundles services.

Implementation of our proposed architecture is based on OSGi platform and is used Consumer and Producer bundles. In this section, we describe the implement of our proposed architecture based on OSGi. First, we present producer consumer binding specified in [6]. That facilitates the explanation of the proposed subscription implementation. Finally fusion/aggregation integration is described.

4.1. Consumer and Producer in OSGI platform

Producer and Consumer are defined in OSGI specifications [6]. A wire is necessary to connect a producer and a consumer. The wireCommand Bundle allows the designer of application to create wires.

Each wire connects one consumer and one producer. A consumer could be connected to several producers and a producer could be connected to several consumers; for each connection, a new wire is used.

Compatibility between the producer and the consumer is check during run-time wire creation process. The verification proposed on OSGi specifications [6] can be based on scopes or types.

The designer can express, in the wire, the condition of sent from the producer. These functionalities are placed into the entity named 'filter'. The designer cans choice filtering type:

- filtering by time (elapsed time),
- filtering by change (update, previous)
- or hysteresis (delta absolute and delta relative).

These five basic filtering conditions are specified in OSGi specifications [6]. The sending condition of the associated wire, expressed by this way, is specified when the wire is created.

To develop our proposed architecture illustrated in fig 4, we use wire, producer and consumer entities to create producers P_1.. P_n and the consumer C_d. OSGi specifications [6] don't provide the *Sub* entity (fig 4). In the next sub-section, we describe the Sub implementation.

4.2. Subscription entity (Sub).

The *Sub* (fig 4) entity is also a producer (C_d side) and consumes data from producers P_i. It is implemented as a bundle in the OSGi platform, which contains producer and consumer interfaces. Wires are created to connect each P_i to the *Sub* entity. Another wire is created to connect the *Sub* entity to the consumer C_d.

4.3. Fusion entity

The fusion process has to provide result from input values. The process could be aggregation type or any type of fusion. We have considered that fusion processes could be used by several *Sub* entities. That explains why fusion process is not present in *Sub* entity. So the fusion entity is implemented as a service bundle. Fusion process parameters are described into an xml file. The sub entity has to use this file entity to know the input fusion parameters. Then each *Sub* entity using the fusion process can use its own parameters for this fusion process.

4.4. Fusion process parameters

There are four types of parameters used in the fusion process. All of these parameters pass through the *Sub*:

- The input process parameters: they are provided by the S*ub* entity during the running process. These parameters are collected by the *Sub* entity from producers P_i.
- Characteristics values can be added to the input process parameters. In our example a weight is link to the data of the producers. These different weights are used for the weighted average method.
- Other parameters, independent of the input *Sub* entity data, used by the fusion process, can be fixed dynamically by the sub entity or by the consumer C_d.
- The output parameters, result of the fusion process.

An XML file provides descriptions of theses parameters. *Sub* entity has to read information (contained in this xml file) in order to be able to bind inputs of fusion process to its parameters' values (x_i, wi) to be sent. All of these parameters are created in the *sub* entity. Example of xml description (see (5) with two inputs):

```
< Aw_bundle >
   < id_service >
      awservice.Awservice
   < /id_service >
   < type >
      < in >
         < x1 > IE < /x1 >
         < x2 > IE < /x2 >
         < w1 val=3> int < /w1 >
         < w2 val=5> int < /w2 >
      < /in >
      < out >
         < r > IE < /r >
      < /out >
   < /type >
   < help >
   …
   < /help >
< /Aw_bundle >
```

x_i and w_i are variables associated which P_i (initialized during the creation process of each wire between P_i and sub entity). In this example, the used type for x_i is IE and is the measurement value of P_i (see (1)). w_i are the weight of each input of *Sub*. The *Sub* entity could provide these parameters

values to the fusion process entity. To perform this, variable names are used and associated to the input type of associated fusion process. These variables are x_1, x_2 and w_1, w_2 in the above example. There are also defined into the Sub entity during creation of wire (between P_i and *Sub* entities). Then x_1 w_1 are associated with the wire1. Wire1 connects P_i. and *Sub*. Example of command to create wire1:

```
wa create P1 Sub val=x1
```

By this way, values produced by producers are propagated to the fusion process (x_i).

At the creation of the *sub* entity, the xml file is read in order to allow the *sub* entity to initialize all of data parameters (except input providers' data). The consumer C_d reads the xml file to know the variable parameters of the fusion process. The fig 5 illustrates the relation between xml file and other entities. The consumer C_d can dynamically modify the parameter values; for example to change weights in (5).

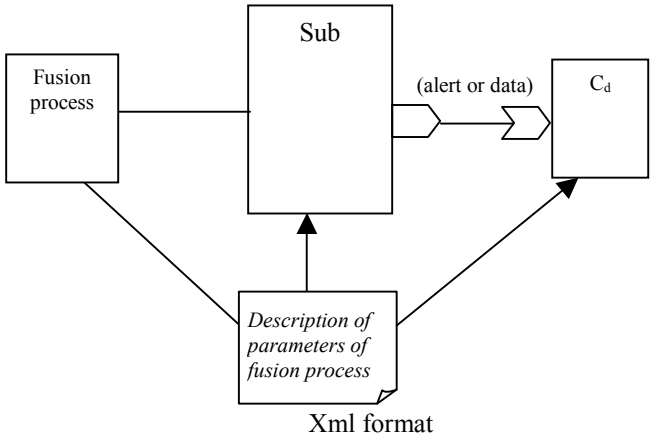

Fig. 5. snapshot of relations between the xml file and the subscription entity

In this example, the output of fusion process could be able to provide the result `r` (IE type).

4.5. Filter

The Filter entity (fig 4) contains condition of sending. This is the case for any wire. For the proposed architecture for subscription process, the filter (fig 4) contains condition of sending of the result of fusion process to the consumer C_d. Filter capabilities specified in [6] not allows the user to easily implemented combination of sending condition, or express condition non-included in [6]. So we propose to use a bundle to implement filter process.

For the example illustrated by (5-6), (6) contains filtering condition. The xml description of filtering constraints is given below:

```
< F_bundle >
   < id_service >
      fservice.Fservice
   < /id_service >
   < type >
      < in >
         < oi > IE < /oi >
         < c1 val=0.05> double < /c1 >
      < /in >
      < out >
         < oo > IE < /oo >
      < /out >
   < /type >
   < help >
   …
   < /help >
< /F_bundle >
```

where

`oi`: data input of filter.

`oo`: data output of filter.

c_1: specific condition used by the filter. In the example (see (6)), filter consists in sending the input data if its value is in confidence interval defined by (6). c_1 is the value of tolerance (0.05 by default in the example).

Like for the fusion process, values can be dynamically modified by the consumer C_d . Same mechanisms are used both for the fusion process or for the filter process.

5. CONCLUSION AND PERSPECTIVE

We propose in this paper architecture and prototype to render possible dynamically the subscription functionality. Our approach facilitates the use of interval confidence representation for measurement information. One interest of our proposition is to express the need of consumer (constraints) and allows him to deal with the system to automatically obtain (subscription) information in accordance to its constraints. Consumer is able to dynamically modify constraints.

REFERENCES

[1] Bloch, G., C. Eugene, M. Robert and C. Humbert, Measurement Evolution: from Sensors to Information Producer, IMEKO TC1 TC7, London, UK, pp 335-341, 8-10 September 1993.

[2] Spoelder, H.J.W., A.H. Ullings, F.C.A. Groen, Virtual Intrumentation: A survey of Standards and their Interrelation, IEEE/IMTC Instrumentation and Measurement Technology Conference, pp 676-681, Ottawa Canada, 19-21 May 1997.

[3] Benoit, E., L. Foulloy and J. Tailland, Automatic Smart Sensors Generation Based on InOMs, 16th IMEKO World Congress, Vol IX, pp 335-340, Vienna, Austria, 25-28 September 2000.

[4] Riviere, J.M., M. Bayart, J.M. Thiriet, A. Bouras, M. Robert, Intelligent instruments: some modelling approaches. Measurement and Control, vol 29, pp. 179 −186, 1996.

[5] Rumbaugh, J., M. Blaha, W Lorensen, F. Eddy, W. Premerlani, Object Oriented Modeling and Design, Prentice-Hall International, New Jersey, 1991

[6] OSGi Service Platform, Service Compendium, Release 4, Version 4.1, OSGi Alliance, 594 pages, May 2005.

[7] Cristina Marin and Mikael Desertot, Sensor Bean : A Component Platform for Sensor-based Services, In proceedings of the 3rd International Workshops on Middleware for Pervasive and Ad-hoc Computing, MPAC'05,,

Grenoble, France, 28-29 November 2005.
[8] Mauris G., Propagation of measurement uncertainty expressed by a possibility distribution with coverage-interval-based semantics in XVIII IMEKO World Congress, CD-ROM , Rio de Janeiro, Brasil, September 2006, 5 pages.